PROCEEDINGS

55. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

13 - 17 September 2010

Crossing Borders within the ABC

Automation,

Biomedical Engineering and

Computer Science

Faculty of
Computer Science and Automation

www.tu-ilmenau.de

TECHNISCHE UNIVERSITÄT
ILMENAU

Impressum
Published by

Publisher:	Rector of the Ilmenau University of Technology Univ.-Prof. Dr. rer. nat. habil. Dr. h. c. Prof. h. c. Peter Scharff
Editor:	Marketing Department (Phone: +49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de) Faculty of Computer Science and Automation (Phone: +49 3677 69-2860) Univ.-Prof. Dr.-Ing. habil. Jens Haueisen
Editorial Deadline:	20. August 2010
Implementation:	Ilmenau University of Technology Felix Böckelmann Philipp Schmidt

USB-Flash-Version.

Publishing House:	Verlag ISLE, Betriebsstätte des ISLE e.V. Werner-von-Siemens-Str. 16 98693 Ilmenau
Production:	CDA Datenträger Albrechts GmbH, 98529 Suhl/Albrechts
Order trough:	Marketing Department (+49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de)

ISBN: 978-3-938843-53-6 (USB-Flash Version)

Online-Version:

Publisher:	Universitätsbibliothek Ilmenau ilmedia Postfach 10 05 65 98684 Ilmenau

Home / Index:
http://www.db-thueringen.de/servlets/DocumentServlet?id=16739

INTEGRATED SOFTWARE DEVELOPMENT OF DISTRIBUTED AUTOMOTIVE SYSTEMS

Christoph Bodenstein, Frank Lohse

TU-Ilmenau
Faculty of Computer Sience and Automation
System and Software Engineering
www.tu-ilmenau.de/sse

ABSTRACT

Model based system design and optimization has a growing impact since decades. Car electronics are one of the main fields where the adoption of complete model based development processes can yield great advances for system design quality and time to market. This paper shows an ongoing research project about an holistic approach for automotive system design by combining existing tools in a process with techniques established by AUTOSAR.

Index Terms— model based system design, embedded systems, autosar, automotive software

1. INTRODUCTION

Over the last years methods of model based software development are widely established for design of complex systems. The design of distributed systems and the step from requirements towards first prototypes is much easier through using executable specifications. With change to model based design cost and time of development decrease while the quality of designed systems grows through test and verification throughout the whole process.

In modern automotive electrical and electronic systems a growing number of software functions have to be mapped on a given amount of electronic control units (ECU). Furthermore each ECU has very limited resources. Model based software engineering is the instrument for handling these issues. The need for a new design concept of automotive electronic systems and networks is shown by actual statistics of car break downs. In 2009 40% of all passenger car break downs were caused by electric or electronic issues (ADAC car break down statistics).

Carmakers threat the lack of an adequate software design process by developing a new standard for automotive software. It‘s called AUTOSAR. This standard has two main targets. First is the decoupling of hard- and software. The application is has no direct access to ECU hardware, so it is runnable on different ECU-types. The second innovation is a complete component based middleware which is one precondition for decoupling hard and software. The configuration of the AUTOSAR middleware is the most difficult task for carmakers changing their software development process to AUTOSAR.

Main issue of the actual research project is the development of an integrated software design process using executable specifications to model the complete car within its environment, optimize the function distribution and generate the AUTOSAR middleware configuration together with the software components [1][2][3][4].

2. IDEA

The research project combines two main tasks. First of all is a unified design process across all abstraction levels of system design. The resulting process should allow to simulate and test the system during early design phases, thus closing the gap between requirements and design decisions that must be based on behavioral properties. This is done by generating executable models of low-level ECU firmware as well as providing a system-level model which can be simulated as standalone or integrated in the expected system environment (mission level approach).

As the simulation of the whole system requires much calculation resources and time but is needed for the system optimization our approach uses a kind of accuracy-adaptive simulation. Performance evaluation results at the mission level are valuable inputs for improved system design on lower levels regarding architecture, functions, and implementations. The proposed design methodology as shown in Fig. 1 and Fig. 2 is influenced by publications of Alberto Sangiovanni-Vincentelli from University of California, Berkeley. The reader may recognize the presented design flow is an altered kind of different actual standard methodologies. As the mapping of SW-functions to hardware is one of the ongoing problems in system design a major part of the intended process deals with that as described in the next section [5][6].

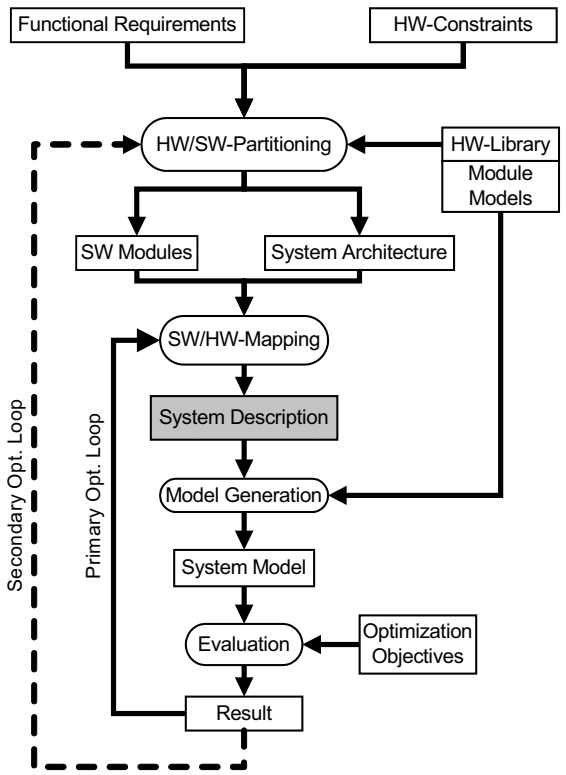

Fig. 1. System Design and Optimization Process

3. ARCHITECTURE OF THE PROPOSED DESIGN PROCESS

The proposed approach of automotive system design is depicted in Fig. 1. A similar approach can, e.g., be found in [6].

It takes the two main factors of system requirements as input:

- **Functional requirements** are given by the initial requirements analysis, and transformed into simple functions to be realized as hardware or software modules.
- **Hardware constraints** like types of micro controllers to be used, memory size or available communication systems like CAN, FlexRay etc.

  Industrial development programs are often constrained to existing architectures and hardware, such that the hardware / software partitioning is restricted, e.g., by existing system designs.
- Non-functional requirements and variants for optimization strategies will be used in later system design phases

The subsequent partitioning maps functions (or software components) to hardware or software. While simple software modules can be created nearly without constraints, only hardware modules from a given model library can be used in our architecture. The modules correspond to existing hardware and are executable.

The Result of the partitioning step are simple software modules and the system architecture including number and type of ECUs or other hardware as well as the communication links in between. At the moment, these design decisions are done by engineers mostly based on experience without algorithmic optimization except for the selection of possible hardware modules.

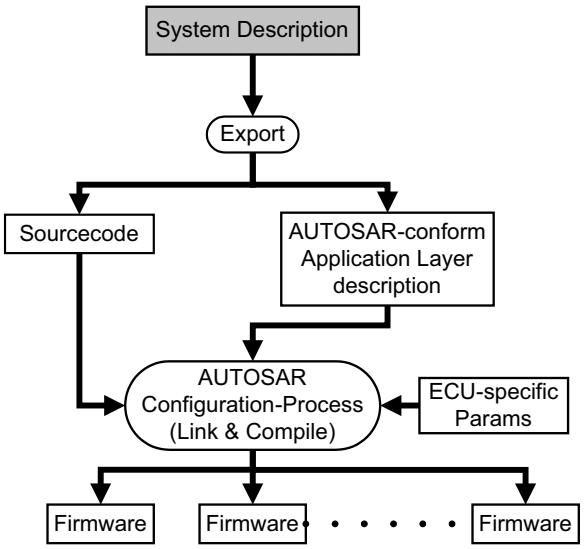

Fig. 2. AUTOSAR System Generation

The inner loop of system optimization regards mapping of software tasks to ECUs. Several optimization algorithms are possible. It is envisaged to integrate simulated annealing, evolutionary algorithms as well as ant-colony algorithms to find an optimal mapping. The fitness function will contain a mixture of multiple objectives including workloads, delays, space, and energy requirements as well as ECU distance as it has also been proposed in [7]. Only fixed network topologies are supported at the moment. In a later development step of our approach it is conceivable that the outer iterative optimization loop (dashed in the figure) will be automated as well, with a heuristic optimization technique for exploring the design space. A first prototyp tool computing an optimal mapping of functions to architectures using heuristic algorithms is described in [8].

A global system description is generated from the mapping information and module library. This description represents an executable specification of the entire system. All necessary information to generate the system model or to export an AUTOSAR conform software configuration is included. Model generation is done by using the corresponding module models from the hardware library. Software module models are generated during runtime of the model generator.

Fig. 2 shows the second part of the design work flow. The resulting system description after optimization is translated into source code for the applications, the software functions, as well as the description of the AUTOSAR application layer. AUTOSAR configuration tools such as Tresos AutoCore[1] or DaVinci Configurator[2] combine the ECU-specific parameter sets for communication and hardware configuration with the application layer description and applications prior to linking and compiling the firmware for all defined ECUs. For debugging and system control reasons, the entire network description is exported by these tools in the proprietary dbc-format which is the de facto standard for CAN networks, defined by Vector Informatik. Future projects integrating other network types like LIN, MOST etc. will export these information using the FIBEX standard as well.

4. TOOLS

To reach the target of a consistent and integrative system design process for automotive software several software tools are evaluated. These tools or parts of it will be combined to get a design process comprising the strengths of every single one. 3 tools are chosen to be used in the future process. All of them are widely known system / software design tools and presented shortly.

One of the most common tools for embedded software design is Matlab Simulink from The MathWorks in combination with dSPACE Targetlink. Simulink allows to model systems based on their internal data flow. It is widely used in control theory and cybernetics. With Real-Time workshop or Targetlink it is possible to generate code for testing, verification and final production use. Therefore these tool combinations enjoyed great popularity in automotive industry for prototyping as development of final production ECU software.

Second important tool is Ptolemy. It was first developed in C++ as an open source tool for modeling, simulation, and design of concurrent, real-time, embedded systems. Version II is written in Java and offers a framework which can be used to build domain specific simulation tools[9]. An example for this is given by the Kepler project, a scientific workflow management system. The Kepler project as a specialized development based on the Ptolemy framework is one inspiration for tailoring the described new design process.

In this context, MLDesigner has to be noticed. MLDesigner is an integrated platform for modeling and analyzing the architecture, function and performance of high level system designs - either as a standalone system or as a system operating in the context of larger systems and scenarios (i.e., missions)[10]. Originally developed at TU-Ilmenau and MLDesign GmbH as an improved system design tool based on the Ptolemy project a wide library of modules for design and simulation of complex systems on different abstraction levels has grown while usage in various projects such as simulation of satellite based positioning system or in avionics projects[11][12].

Because modeling on high level (mission level) is the focus of tools like Ptolemy and MLDesigner code generation from this tools is not very sophisticated but an ongoing part of development in different working groups[13][9].

The strength and preferred fields of application of the presented tools are depicted in figure 3. As shown the main field of usage for MLDesigner and Ptolemy is the system design on abstract levels like architecture system or mission level. Matlab Simulink is mostly preferred for graphical implementation of control algorithms and functions such as used on embedded systems in automotive. Especially the generation of code for final production makes it very useful as part of the tool chain in the targeted design process.

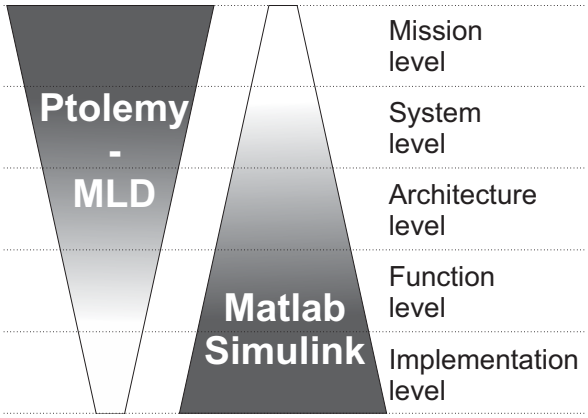

Fig. 3. Overview: Levels of system design & tool classification

So it is obvious to combine tools like Ptolemy or MLDesigner for high level system design with Matlab Simulink for function modeling. In this context high level means the mission which describes the car within its environment. The system level represents the set of different ECUs within a car network including the communication description on abstract level. The mapping of functions or software components on ECUs including the inter-function communication between ECUs is part of the architecture level. For simplification each ECUs is modeled as a single-processor system. The exact model on these levels is necessary for two reasons. The simulation on different levels of abstraction offers many fields of optimization. The mapping of functions to ECUs requires a very detailed model of the given hardware and the underlying operating system to simulate the execution time of functions. The exact model of communication system is not required at this optimization steps. Therefore queues and time outs can be used

[1]Tresos AutoCore: Elektrobit Automotive GmbH
[2]DaVinci Configurator: Vector Informatik GmbH

to simulate the network arbitration. Concrete sample data about execution and communication time can be obtained by validating the system model on real hardware. The generation of production code from Matlab Simulink in Combination with dSPACE Targetlink is a key feature in this step. In turn the gained information from the running prototype system is used in higher level simulation to reduce computing time by using simple abstractions such as timers and queues instead of exact function models.

After all, the decision which part of the system will be simulated on which abstraction level to find a balance between simulation time for optimization cycles and accuracy of the system simulation results has to be made by a so called *simulation mediator* which is one part of the actual research project[14].

The end of the announced design process is marked by an AUTOSAR-conform export of all system information from the model. While the software components and functions are modeled with Matlab Simulink the generation of C-Code therefore is the smallest part at this point. Much more complex is the information about the system and software architecture, communication specifications and operating system configurations. Actually gathering this information and configuring the ECUs basic software is the main challenge for companies migrating their software design process to AUTOSAR.

The export of AUTOSAR system config information consists of application layer description, communication specifications and required data for AUTOSAR OS configuration. In this context application layer description means mapping of functions to software components (SWC), SWC to AUTOSAR runnables and the link of module ports to real signals (ECU internal or network signals).

5. CONCLUSION

The necessity of a unified and all-embracing design process for automotive electric/electronic systems is generally accepted. It is driven by the growing number of ECUs and software functions, increasing complexity of in car networks and the need for improved software quality especially with focus on x-by-wire technologies.

With our actual research project we aim at the definition of these future system design process as well as its prototypical implementation.

6. REFERENCES

[1] Jon Friedman, "Model-based design helps aerospace engineers," *Design News*, June 2009.

[2] Nicolas Navet and Francoise Simonot-Lion, *Automotive Embedded Systems Handbook*, CRC Press, Inc., Boca Raton, FL, USA, 2008.

[3] Jason Ghidella and Jon Friedman, "Model-based design streamlines development of body electronics systems," *Auto electronics*, November / December 2005.

[4] Amos Albert, "Comparison of event-triggered and time-triggered concepts with regard to distributed control systems," in *Proceedings of Embedded World*, 2004.

[5] Harald Heinecke, "AUTomotive Open System ARchitecture an industry-wide initiative to manage the complexity of emerging automotive E/E-architectures," in *Convergence 2004, International Congress on Transportation Electronics*, 2004.

[6] Alberto Sangiovanni-Vincentelli, "Electronic-system design in the automobile industry," *Micro, IEEE*, vol. 23, no. 3, pp. 8 – 18, May-June 2003.

[7] Bernd Hardung, *Optimisation of the Allocation of Functions in Vehicle Networks*, Ph.D. thesis, Universität Erlangen-Nürnberg, 2006.

[8] Frank Lohse, Volker Zerbe, and Thomas Luetzelberger, "Architecture analysis and optimization of reconfigurable, complex systems," in *Proc. 14th IEEE Int. Conf. on Intelligent Engineering Systems (INES 2010)*, Gran Canaria, Spain, May 2010.

[9] "http://ptolemy.eecs.berkeley.edu," 2009.

[10] "http://www.mldesigner.com," 2010.

[11] Ankur Agarwal, Cyril-Daniel Iskander, Ravi Shankar, and Georgina Hamza-Lup, "System-level modeling environment: MLDesigner," in *IEEE Int. Systems Conf. (SysCon 2008)*, Montreal, Canada, April 2008.

[12] Volker Zerbe, Mario Schulz, Armin Zimmermann, and Stephan Marwedel, "Model-based evaluation of avionics maintenance and logistics," in *Proceedings of IEEE SMC 2010*, 2010.

[13] Holger Rath, "ANSI C code synthesis of MLDesigner finite state machines," M.S. thesis, TU-Ilmenau, 2003.

[14] M. Radetzki and R. Salimi Khaligh, "Accuracy-adaptive simulation of transaction level models," in *Proc. Design, Automation and Test in Europe (DATE)*. 2008, pp. 788–791, IEEE Computer Society.